Student Name

CS 340 – Operating Systems

Dr. Simina Fluture

Homework #5

***Part 1 – Unix-Processes***

*Section A*

18.) *Foreground And Background*

Prompt:

[ENTERYOURVENUSNAMEHERE@venus ~]$ ls -l > dirlist &

[2] 22029

[1] Done ls --color=tty -l >dirlist

Explanation:

This prompt makes the dirlist run in the background because of the “&”.

Prompt:

[ENTERYOURVENUSNAMEHERE@venus ~]$ fg [%jobid]

-bash: fg: [%jobid]: no such job

[2]+ Done ls --color=tty -l >dirlist

Explanation:

This prompt takes all the jobs in the background and brings them to the foreground and completes their processes.

*Suspend*

Prompt and Display:

[ENTERYOURVENUSNAMEHERE@venus ~]$ ps

PID TTY TIME CMD

21993 pts/10 00:00:00 bash

22107 pts/10 00:00:00 ps

[ENTERYOURVENUSNAMEHERE@venus ~]$ vi

-- INSERT --

#include <stdio.h>

#define SIZE 100

main (int argc, char \*argv[ ])

{

...

(^Z)

[1]+ Stopped vim

[ENTERYOURVENUSNAMEHERE@venus ~]$ ps

PID TTY TIME CMD

21993 pts/10 00:00:00 bash

22120 pts/10 00:00:00 vim

22161 pts/10 00:00:00 ps

[ENTERYOURVENUSNAMEHERE@venus ~]$ jobs

[1]+ Stopped vim

Explanation:

In the prompts above basically the vim process was suspended after hitting Ctrl+Z. It was then see in the process states (ps) and then in jobs.

19.) *Nice*

Prompt:

[ENTERYOURVENUSNAMEHERE@venus ~]$ man nice

Explanation:

Run COMMAND with an adjusted scheduling priority. With no COMMAND, print the current scheduling priority. ADJUST is 10 by default. Range goes from -20 (highest priority) to 19 (lowest).

20.) *Abnormal termination of processes*

Prompt And Display:

[ENTERYOURVENUSNAMEHERE@venus ~]$ ps

PID TTY TIME CMD

21993 pts/10 00:00:00 bash

22120 pts/10 00:00:00 vim

22301 pts/10 00:00:00 man

22304 pts/10 00:00:00 sh

22305 pts/10 00:00:00 sh

22310 pts/10 00:00:00 less

22765 pts/10 00:00:00 ps

[ENTERYOURVENUSNAMEHERE@venus ~]$ kill -15 22120

[ENTERYOURVENUSNAMEHERE@venus ~]$ ps

PID TTY TIME CMD

21993 pts/10 00:00:00 bash

22301 pts/10 00:00:00 man

22304 pts/10 00:00:00 sh

22305 pts/10 00:00:00 sh

22310 pts/10 00:00:00 less

22768 pts/10 00:00:00 ps

[1]- Killed vim

Explanation:

In the prompts above we see that the vim process is terminated abnormally by using the kill command.

*Section B*

1.) Prompt:

[ENTERYOURVENUSNAMEHERE@venus Projects]$ date& who; whoami; uname; echo Hello, World&

Displayed:

[1] 18871

Mon Apr 5 17:30:50 EDT 2010

wacu7584 pts/1 Apr 5 14:45 (cpe-24-193-96-77.nyc.res.rr.com)

leda4410 pts/4 Apr 5 13:43 (pool-71-167-78-49.nycmny.east.verizon.net)

leda4410 pts/6 Apr 5 17:19 (pool-72-89-197-224.nycmny.east.verizon.net)

wobr4396 pts/8 Apr 5 16:42 (pool-98-116-187-124.nycmny.fios.verizon.net)

ENTERYOURVENUSNAMEHERE pts/9 Apr 5 17:27 (user-12hduqc.cable.mindspring.com)

[1]+ Done date

ENTERYOURVENUSNAMEHERE

Linux

[1] 18875

Hello, World

2.) Prompt:

[ENTERYOURVENUSNAMEHERE@venus ~]$ man nice

Explanation:

The default for Adjust is 10.

***Part 2 – Unix-Processes***

Sources: <http://www2.cs.uregina.ca/~hamilton/courses/330/notes/scheduling/scheduling.html>

<http://en.wikipedia.org/wiki/Scheduling_(computing)#Mac_OS>

Windows XP Scheduling Algorithm

* Preemptive priority scheduling algorithm (quantum-based).
* Instead of processes, threads are scheduled
* The algorithm is made up of multiple queues -> Multiple Feedback Queue
* A thread is preempted because:
  + Higher- priorty thread becomes ready
  + A thread terminates
  + The quantum time is exhausted
  + Thread does a blocking system call.
* The highest priority in Windows XP is 31 and the lowest is 0.
* A running thread is always at the highest priority.
* An idle thread is run if no other thread is being run.

Unix/Linux Scheduling Algorithm

* Preemptive priority algorithm with 140 priority values.
* There is a static priority in a range of -20 to 19 where -20 is the highest.
* Runqueue
  + Active Priority Array
  + Expired Priority Array
* The active priority array is emptied first. When it is empty the expired priority array becomes the active priority array and visa versa.
* The highest priority processes are granted longer quanta and the lower priority processes are granted shorter quanta.

MacOS Scheduling Algorithm

* Uses cooperative scheduling. One process controls multiple cooperative threads.
* The kernel uses round-robin scheduling and each process has its own copy of the thread manager to schedule each thread.
* The kernel then uses a preemptive scheduling time to schedule all tasks to have processor time.
* Mach threads are linked to its own separate process. If the threads are cooperative then only one can run at a time.
* Threads give up their rights to the processor for other processes to run.